# CERTIFICATE OF VERIFICATION

I, David H. Owens,
of Nisihshinbashi 3 Mori Bldg. 4F, 1-4-10 Nishisinbashi, Minato-ku, Tokyo, Japan

state that the attached document is a true and complete translation to the best of my knowledge of Japanese Patent Application No. 2003-086828.

Dated this 2nd day of October, 2006.

Signature David H Owens

David H. Owens

| | |
|---|---|
| 11004a | File Access Control Memory |
| 11007a | Inter-CPU Communications Circuit |
| 11008a | Disk Array Control CPU |
| 110011a | Data Transfer Control Circuit |
| 11009a | Disk Array Control Memory |
| 110010a | FC Controller |
| 1a | Storage Device (STR 1) |
| 170a | Disk Pool 0 (SATA Disk Pool) |

Fig. 14

| | |
|---|---|
| 400 | NAS Host 0 |
| 1 | Storage Device (STR 0) |
| 11012b | FC Controller |
| 11008b | Disk Array Control CPU |
| 11011b | Data Transfer Control Circuit |
| 11009b | Disk Array Control Memory |
| 11010b | FC Controller |
| 1b | Storage Device (STR 2) |
| 1700b | Disk Pool 0 (SATA Disk Pool) |

Fig. 15

| | |
|---|---|
| 400 | NAS Host 0 |
| 401 | NAS Host 1 |
| 402 | NAS Host 2 |
| 403 | NAS Host 3 |
| 500 | SAN Host 0 |
| 501 | SAN Host 1 |
| 502 | SAN Host 2 |
| 170 | Disk Pool 0 (FC Disk Pool) |
| 171 | Disk Pool 1 (SATA Disk Pool) |





| | |
|---|---|
| 1c | Storage Device (STR 3) |
| 170a | Disk Pool 1 (SATA Disk Pool) |
| 1a | Storage Device (STR 1) |
| 170b | Disk Pool 2 (SATA Disk Pool) |
| 1b | Storage Device (STR 2) |

Fig. 16

| | |
|---|---|
| 7201 | LAN Controller |
| 7202 | File Access Control CPU |
| 7203 | File Access Control Memory |
| 7204 | SW Node Controller |

Fig. 17

| | |
|---|---|
| 7301 | FC Controller |
| 7302 | SW Node Controller |

Fig. 18

| | |
|---|---|
| 7401 | LAN Controller |
| 7402 | SW Node Controller |

Fig. 19

| | |
|---|---|
| 7501 | SW Node Controller |
| 7502 | Disk Array Control CPU |
| 7505 | Data Transfer Control Circuit |
| 7503 | Disk Array Control Memory |
| 7504 | FC Controller |

Number of Blocks
Link Destination Node Name
Link Destination FS Name
Link Destination Filename
Link Destination File Handler
Buffer Management Table
1100438 File Property Information Management Table
1100437 Buffer Management Table
Hash Link
Queue Link
Flag
Device Number
Block Number
Number of Bytes
Buffer Size
Buffer Pointer

Fig. 11 (from main title to subtitle; top to bottom; left to right)
1100438 File Property Information Management Table
Property Information Type
Category
Attribute
Content
Static Property Information
File Information
Policy
File Type
Application
Date Created
Owner
Access Identifier
Initial Storage Class
Asset Value Type
Life Cycle Model
Migration Plan
Document





XYZ Word
Not designated
Important
Model 1
Plan 1
Dynamic Property Information
Access Information
Life Cycle stage Information
Time Stamp
Access Count
Read Count
Write Count
Read Size
Write Size
Read Sequential Count
Write Sequential Count
Current Life Cycle
Current Storage Class
Reference Stage
NearLine

Fig. 12
110435 File Storage Management Table
File Property Information Management
Link Destination Node Name
Link Destination FS Name
Link Destination Filename
Buffer Management Table
1100438 File Property Information Management Table

Fig. 13
400 NAS Host 0
1 Storage Device
(STR 0)
11002a LAN Controller
11001a File Access Control CPU

Fig. 1

400 NAS Host 0
401 NAS Host 1
402 NAS Host 2
403 NAS Host 3
404 NAS Host 4
405 NAS Host 5
500 SAN Host 0
501 SAN Host 1
502 SAN Host 2
131 Disk Pool Management Table
170 Disk Pool 0
(FC Disk Pool)
171 Disk Pool 1
(SATA Disk Pool)
1 Storage Device (STR0)

Fig. 4

11002 LAN Controller
11001 File Access Control CPU
11004 File Access Control Memory
11007 Inter-CPU Communications Circuit
11008 Disk Array Control CPU
11009 Disk Array Control Memory
11005 SM I/F Control Circuit
11006 CM I/F Control Circuit

Fig. 5

1100431 File open processing section
1100432 Request processing section
1100433 File storage management section
1100434 Buffer management section
110043A Migration management section
1100435 File storage management table
1100436 Filename management table
1100437 Buffer management table





1100438 File property information management table
1100439 Storage class management table
110043 File System Program
11004 File Access Control Memory
110045 Volume Control Program
110042 TCP/IP Program
110044 Network File System Program
110041 LAN Controller Driver Program
110046 Inter-CPU Communications Driver Program
110040 Operating System Program

Fig. 6

110091 Disk Array Control Program
110093 Inter-CPU Communications Driver Program
110092 Disk Pool Management Program
110094 Cache Control Program
110095 DKA Communications Driver Program
11009 Disk Array Control Program
110090 Operating System Program

Fig. 7

170 FC Disk Pool
171 SATA Disk Pool

Fig. 8

Storage Class Management Table

Fig. 9

1100436 Filename Management Table
Filename
File Handler
1100435 File Storage Management Table

Fig. 10

1100435 File Storage Management Table
File Property Information Management Table

170: disk pool

400: NAS host

500: SAN host

1100: NAS channel adapter

1110: Fiber Channel node





[Document Name] ABSTRACT

[Abstract]

[Problem]

Conventionally, since the storage hierarchies were realized in the programs of the hosts and also realized on a per-logical disk basis, dependence on the computer was high, thus there was a limitation in simplification of the flexible system configuration and management. Moreover, there was no consideration for optical placement of the files in accordance with the file properties.

[Means of Solution]

A storage device is provided with a file I/O interface control device and a plurality of disk pools. The file I/O interface control device sets one of a plurality of storage hierarchies defining storage classes, respectively, for each of LUs within the disk pools, thereby forming a file system in each of the LUs. The file I/O interface control device migrates at least one of the files from one of the LUs to another one of the LUs of an optimal storage class, based on static properties and dynamic properties of each file.

[Selected Drawing] Fig. 7

[Brief Description of the Drawings]

Fig. 1 is a diagram of a configuration example of a computer system to which the present invention is applied.

Fig. 2 is a diagram of an example of the exterior appearance of a storage device.

Fig. 3 is a diagram of an example of the exterior appearance of an adapter board.

Fig. 4 is a diagram of a configuration example of a NAS channel adapter.

Fig. 5 is a diagram of an example of programs stored in a file system control memory.

Fig. 6 is a diagram of an example of programs stored in a disk array control memory.

Fig. 7 is a diagram of an example of the relationship among disk pools, LUs and file systems.

Fig. 8 is a diagram of an example of a storage class management table.

Fig. 9 is a diagram of an example of a filename management table.

Fig. 10 is a diagram of an example of a file storage management table and a buffer management table.

Fig. 11 is a diagram of an example of a file property information management table.

Fig. 12 is a diagram of an example of a file storage management





table.

Fig. 13 is a diagram of a second configuration example of a system to which the present invention is applied.

Fig. 14 is a diagram of a third configuration example of the system to which the present invention is applied.

Fig. 15 is a diagram of a fourth configuration example of the system to which the present invention is applied.

Fig. 16 is a diagram of a configuration example of a NAS node.

Fig. 17 is a diagram of a configuration example of a Fiber Channel node.

Fig. 18 is a diagram of a configuration example of an IP node.

Fig. 19 is a diagram of a configuration example of a disk array node.

[Explanation of Reference Numerals]

1: storage device

13: shared memory

14: cache memory

15: shared memory controller

16: cache memory controller

20: LAN

21: LAN

30: SAN

35: SAN

120: disk control adapter

and control information to and from the STR2. Through the configuration and processing procedure described above, as in the third embodiment, a file-based storage hierarchy that utilizes a file system that is constructed in an external storage device STR2, and managed by the STR3 can be realized.

[0159]

According to the present embodiment, the storage device STR3 behaves as if it were a central control controller for constructing a hierarchy storage system and various types of storage devices can be connected internally and externally to the storage device STR3; consequently, an extremely flexible, scalable and large-scale hierarchical storage system can be constructed. Furthermore, due to the fact that disks and other storage devices can be connected internally and externally to the storage device STR3 as nodes on the SW of the storage device STR3, high-speed data transfer becomes possible.

(7) Other Applications

Although file transfer methods and storage devices that execute hierarchical migration processing of files based on the file's data life cycle stages have been described in the first through fourth embodiments, files can be transferred based on other standards and a plurality of standards can be combined. Possible standards other than the data life cycle stage include a file's access property and an LU's used capacity. Also, transfer of files can be controlled by providing a migration plan based on the file's access property or the LU's used capacity.





[0160]

Examples of migration plans based on a file's access property include a plan to re-transfer a file into a storage class one class higher in the hierarchy, once the access frequency of the file exceeds a certain level, or a plan that provides a storage class specialized for sequential accesses and that transfers a file into this storage class once the sequential access frequency to the file exceeds a certain level.

[0161]

Examples of migration plans based on an LU's used capacity include a plan to transfer the file with low access frequency that is stored in an LU or the file having a long elapsed time since its date created, to a storage class one class lower in the hierarchy, even if its current life cycle stage has not changed, once the used capacity of the LU exceeds a certain level.

[0162]

The file property information management table 1100438 in the above embodiments manages access information for each file as dynamic properties. The storage class management table also manages the total capacity and used capacity of each LU. By utilizing such information, migration plans described above can be readily realized.

[0163]

A hierarchical storage control according to the property of a file can be realized through a processing within a storage device and without being dependent on the host computers.

class management table 1100439. Subsequent steps are the same as in the processing procedure in the second embodiment.

[0154]

In terms of issuing a disk I/O command, the SW node driver program stored in the file access control memory 7203 of the NNODE is executed by the file access control CPU 7202, which causes a disk I/O command to be issued from the NNODE via the SW node to the INODE 740 connected to the storage device STR1 (1a) that is provided with the LU that is the subject of access. Based on the I/O command received, the INODE 740 issues to the storage device STR1 (1a) a disk I/O command for performing a file access, as well as sends and receives actual data of the file and control information to and from the STR1 (1a).

[0155]

The INODEs 74x have no involvement whatsoever in the file control information and operate simply as gateways of an IP network. In such a case, a hierarchical storage configuration without any interference from other devices, such as NAS hosts, can be realized. Of course, the divergent storage device STR1 (1a) can be connected to the LAN 20, to which the NNODE 720 is connected, as in the second embodiment.

[0156]

Through the configuration and processing described above, a file-based storage hierarchy that uses storage pools of an external divergent storage device, as in the second embodiment, can be realized.





[0157]

Furthermore, as in the third embodiment, the SAN-type divergent storage device STR2 (1b), which is a storage device provided with block I/O interfaces, could be connected externally to the storage device STR3, which would result in a configuration of a storage hierarchy. When the file system program 110043 stored in the file access control memory of the NNODE 72x is executed by the file access control CPU, the NNODE queries the FNODEs 73x whether there is a SAN-type divergent storage device connected to the FNODEs 73x. If there is a divergent storage device connected, the NNODE recognizes remote LUs of the divergent storage device, based on the contents of the response from the FNODEs 73x to the query, and constructs local file systems in the remote LUs. The NNODE then defines a storage class for each of the remote LUs and the local file systems, and registers and manages information concerning those LUs in the storage class management table 1100439. Subsequently the steps that are the same as in the third embodiment are executed.

[0158]

In terms of issuing a disk I/O command, the SW node driver program is executed by the file access control CPU 7202, which causes a disk I/O command to be issued from the NNODE via the SW node to the FNODE 732 connected to the storage device STR2 (1b), which is provided with the LU that is the subject of access. The FNODE 732 issues to the storage device STR2 a disk I/O command, as well as sends and receives data

storage device are described below.

[0149]

In the present embodiment, a hierarchical storage control inside the storage device STR3 can be executed using a procedure similar to that in the first embodiment. A file system program 110043 stored in a file access control memory of the NNODE 72x is equipped with a storage class management table 1100439 for managing usable LUs, and can recognize disk pools and LUs managed by the DNODEs 75x by referring to the storage class management table. However, unlike the first embodiment, there is no SM 13 for storing shared information; consequently, the NNODE 72x must query all DNODEs 75x in advance to specify a usable LU and register it in the storage class management table. Of course, an SM node for connecting with an SM can be provided for connection with the SW in the present embodiment as well, so that the storage class management table can consist of information stored in the SM, as in the first embodiment.

[0150]

When the NNODE 72x specifies a usable disk pool and an LU, creates the storage class management table 1100439, and defines a storage class, a processing similar to that in the first embodiment can be applied subsequently to execute a hierarchical storage control within the storage device STR3 (1c), i.e., a hierarchical storage control using LUs set in the disk pool 0 and the disk pool 1.

[0151]





In terms of issuing a disk I/O command, an SW node driver program stored in the file access control memory 7203 of the NNODE is executed by the file access control CPU 7202, which causes a disk I/O command to be issued via the SW node to the DNODE 750 that manages the LU that is the subject of access.

[0152]

Through the configuration and processing described above, a system in which a file-based storage hierarchy is constructed within the storage device STR3, as in the first embodiment, can be realized.

[0153]

Furthermore, as in the second embodiment, the NAS-type divergent storage device STR1 (1a) provided with file I/O interfaces can be connected externally to the storage device STR3, which would result in a configuration of a storage hierarchy. When the file system program 110043 stored in the file access control memory 7203 of the NNODE 72x is executed by the file access control CPU, the file system program 110043 queries the INODE 74x whether there is a NAS-type divergent storage device connected to the INODE 74x; if there is a divergent storage device, the file system program 110043 obtains from the divergent storage device the information for identifying remote LUs and remote file systems that exist in the divergent storage device. Through a control by the file access control CPU, a storage class is defined for each of the remote LUs and the remote file systems, and information concerning the LUs is registered and managed in the storage

converted frames to and from other nodes, such as the DNODEs.

[0144]

The FNODE 73x operates as an initiator device and based on disk I/O commands received from the NNODEs or other FNODEs can send I/O commands to other storage devices connected externally to the storage device STR3. For example, based on commands received from the NNODEs or other FNODEs of the storage device STR3, the FNODE2 and the FNODE3 in Fig. 15 can send I/O commands to the divergent storage device STR2 (1b) externally connected to the storage device STR3. In this case, the FNODE2 and the FNODE3 appear to be operating as host computers from the perspective of the STR2.

[0145]

Although only the FC controller 7301 and the SW node controller 7302 are shown in Fig. 17 for the sake of simplification, a CPU can be mounted on the FNODEs in order to perform target processing, initiator processing or internal frame generation processing.

[0146]

Note that, by installing an iSCSI controller instead of the FC controller 7301, a node that controls iSCSI can be configured; by connecting such a node to the SW 71, an IP SAN can be configured.

[0147]

(4) Example of Configuration of INODE (Fig. 18)

Fig. 18 is a diagram of an example of the configuration of the





INODE. The INODE 740 has a configuration in which an SW node controller 7402 is connected to a LAN controller 7401 that the NCTL0 (1100a) in Fig. 13 has, and an connect with the SW 71 via the SW node controller. The INODEs are provided on the storage device STR3 (1c) in order to connect the external NAS-type storage device STR1a to the STR3.

[0148]

(5) Example of Configuration of DNODE (Fig. 19)

Fig. 19 is a diagram of an example of the configuration of the DNODE. In the DNODE 750 the FC controller 11012b of FCTL 1100b shown in Fig. 14 is removed and replaced with an SW node controller 7501. The DNODE 750 goes into operation when it receives a disk I/O command from one of the NNODEs or FNODEs via the SW 71; as a result, a section 1d outlined by a broken line in Fig. 15 operates as if it were the independent storage device STR2 in Fig. 14. In the present embodiment; the DNODE0 (750) and the DNODE1 (751) operate as a pair of redundant controllers. Having redundant DNODEs is similar to the configuration of the storage device STR2 in Fig. 14, where there are also redundant FCTLs.

(6) Migration Processing of Files

The present embodiment only differs from the first, second and third embodiments in its configuration of the storage device, and its processing procedure for executing a hierarchical storage control is similar to that in the first, second and third embodiments; accordingly, only those parts that differ in the operation as a result of differences in the configuration of the

switch, NNODEs (72x) are NAS nodes each provided with a file I/O control mechanism to connect with a LAN, FNODEs (73x) are FC nodes each provided with a block I/O control mechanism to connect with a SAN, INODEs (74x) are IP nodes each provided with an IP network control mechanism to connect with an IP network, and DNODEs (75x) are Disk Control nodes each provided with a disk control mechanism to connect with a disk pool. To the switch SW 71 are connected one or more NNODEs 72x, one or more FNODEs 73x, one or more INODEs 74x and one or more DNODEs 75x. A node to control iSCSI can be connected to the switch SW 71 to form an IP SAN. The node to control the iSCSI would have functions and a configuration similar to those of the FNODE.

[0139]

The DNODE0 and the DNODE1 are connected to and control two types of disk pools, a disk pool 0 and a disk pool 1 of an FC disk pool 170 and a SATA disk pool 171.

[0140]

The INODE0 and the INODE1 are connected to a NAS-type divergent storage device STR1 (1a), which exists externally on the storage device STR3 and is a storage device provided with file I/O interfaces described in the second embodiment. The FNODE2 and the FNODE3 are connected to a SAN-type divergent storage device STR2 (1b), which exists externally on the storage device STR3 and is a storage device provided with block I/O interfaces described in the third embodiment.





[0141]

(2) Example of Configuration of the NNODE (Fig. 16)

Fig. 16 is a diagram of an example of the configuration of the NNODE. The NNODE 720 is equivalent to the CHN 1100 shown in Fig. 4 with the inter-CPU communications circuit 11007 and components below it removed and replaced by an SW node controller 7404. Other components are the same as in the CHN in terms of configuration and function.

[0142]

The SW node controller 7204 is a controller circuit for connecting with the SW 71; it forms commands, data and control information in internal frame formats that are sent and received within the storage device STR3 (1c) and sent as disk I/O to other nodes such as the DNODEs.

[0143]

(3) Example of Configuration of the FNODE (Fig. 17)

Fig. 17 is a diagram of an example of the configuration of the FNODE. The FNODE 730 has a configuration in which an SW node controller 7302 is connected to the FC controller 11012b of the FCTL 1100b in Fig. 14, which makes the FNODE 730 capable of connecting with the SW 71 via the SW node controller. An FC controller 7301 operates as a target device and sends and receives frames of commands, data and control information to and from the SAN. The SW node controller 7302 converts frames sent or received by the FC controller into internal frame configurations of the storage device STR3 (1c) and sends or receives the

place on the local file system according to the first embodiment in terms of the file open processing, write processing and migration processing, except for the fact that the processing is executed with the awareness that the link destination node of the file abc.doc (i.e., the storage device in which the actual data of the file abc.doc is stored) is the STR2.

[0136]

However, unlike the first embodiment in which a file is transferred to an LU that exists in the primary storage device STR0, data of a file is transferred to an LU that is in the other storage device STR2 according to the present embodiment, which results in input/output processing to and from disks that is different from the first embodiment. While the DKA 12x of the STR0 controlled the input/output processing to and from disks in the first embodiment, the CHF1 (1111) of the STR0 controls the processing according to the configuration of the present embodiment. For this reason, a CHF communications driver program 110096 is stored in a disk array control memory 11009 of the CHN0. A CHF communications driver section is realized by having a disk array control CPU 11008 execute the CHF communications driver program. The CHF communications driver section sends a disk input/output command (hereinafter called an I/O command) to the SM 13. Address information indicating storage positions of the data is included in the I/O command. The CHF1 (1111) receives the I/O command via the SM 13 and, based on the I/O command received, issues an I/O command to the storage device 1b (STR2) via the SAN 35. The I/O





command issued by the CHF1 (1111) includes address information indicating the data storage positions within the storage device 1b (STR2). The storage device 1b (STR2) processes the I/O command received from the CHF1 (1111) according to the same procedure applied when a disk I/O command is received from a normal host. In other words, the CHF1 of the STR0 is recognized as a host from the perspective of the STR2.

[0137]

According to the present embodiment, the disk pool of the divergent storage device STR2 provided with the block I/O interface can be treated as one of the disk pools of the storage device STR0, and a file system managed by the STR0 can be constructed on the LU that exists in the disk pool of the STR2. Furthermore, due to the fact that files stored in the LU of the STR0 can be migrated to the LU within the STR2, a flexible storage hierarchy with superior cost effectiveness can be constructed.

Embodiment 4:

(1) Example of System Configuration (Fig. 15)

The following is a description of the fourth embodiment. The present embodiment differs from preceding embodiments in its configuration of the storage device

[0138]

Fig. 15 is a diagram of an example of the system configuration according to the present embodiment. A storage device STR3 (1c) is provided with a DKC 70 and disk pools. In the DKC 70, an SW 71 is a

(1100) of the STR0 executes a file system program, the disk pool management table 131 is referred to and the information concerning the LU is copied from the disk pool management table 131 to a storage class management table 1100451 in a file access control memory.

[0131]

As in the second embodiment, it is assumed that a migration management section 110043A of the CHN0 (1100) of the STR0 has decided to migrate a file abc.doc from a NearLine Storage class to the Archive Storage class, and the following is a description of the migration processing of the file executed based on this assumption.

[0132]

The migration management section 110043A of the STR0 refers to a storage class management table 1100439, selects the LU4 that falls into the "Archive Storage" class, and decides to transfer the file abc.doc to the LU4. The LU4 has attributes of "STR2 (i.e., the other storage device 1b)" as Storage Node, "SATA disk pool" as Disk Pool #, and "Remote Block" as its LU type.

[0133]

Unlike the second embodiment, since the LU type of the LU4 is "Block" type, there is no file system in the LU4. For this reason, the file system program stored in the CHN0 (1100) constructs a local file system LFS4 in the LU4. Due to the fact that the disk pool in which the LU4 is set resides in the other storage device STR2 from the perspective of the STR0, it





is therefore a "remote" disk pool and the LU4 is a remote LU; however, since the file system LFS4 set in the LU4 is to be controlled by the CHN0 (1100), the file system LFS4 is managed as a local file system.

[0134]

Due to the fact that the LFS4 is to be managed as a local file system and the LU4 in which the LFS4 is constructed is an LU that exists in the block type, divergent storage device, a file storage management table is treated differently in the present embodiment compared to its treatment in the first and the second embodiments. In other words, a file storage management section 1100433 of the CHN0 (1100) of the STR0 assigns "STR2" as the link destination node name, "LFS4" as the link destination FS name, and a STR2-FILE00001 as the link destination filename, and sets these in a file storage management table for the file abc.doc. Note that, since the file abc.doc has already being migrated to the LU2 under the filename FILE00001, the CHN0 (1100) can alternatively set the assigned link destination node name, the link destination FS name and the link destination filename in the file storage management table for the file FILE00001 in the LFS2. Since the STR2, in which the LU4 actually exists, does not execute the file access control as described earlier, a file storage management table for the STR2-FILE00001 is not created in the STR2.

[0135]

The processing that takes place by executing the file system program 11004 of the CHN0 (1100) is the same as the processing that takes

device STR1 (1a) according to the second embodiment.

[0128]

The SAN 35 is a dedicated network for connecting the storage device STR0 (1) to the storage device STR2 (1b), and SAN hosts are not connected to the SAN 35. For the sake of simplification, let us assume that in the present embodiment no SAN hosts are connected to the SAN 35, which is the network to connect the storage devices, and that there is only one network that connects the storage devices. However, SAN hosts can be connected to the SAN 35 and a plurality of networks for connecting the storage devices can be provided to improve fault tolerance.

[0129]

In the present embodiment, the storage device STR2 (1b) is under the control of the storage device STR0 (1), and file accesses from a NAS host 0 reaches the storage device STR2 (1b) via the storage device STR0 (1). Such a configuration is hereinafter called a "connection of divergent storage devices."

(2) Migration Processing of File to the Other Storage Device

Next, a description will be made as to the processing for migrating a file stored in the STR0 to the STR2, with emphasis on the difference between this processing and the processing according to the second embodiment.

[0130]

A CHF1 (1111) of the storage device STR0 (1) recognizes that the





storage device STR2 (1b), which is a divergent storage device, is connected to the SAN 35. The CHF1 (1111) becomes an initiator and issues a command to collect information and thereby recognizes that the STR2 (1b) is connected to the SAN 35. The CHF1 (1111) treats storage regions of the STR2 as if they were a disk pool within the primary storage device according to the first embodiment. A CHN0 (1110) can use the disk pool via the CHF1 (1111). The management method of the disk pool will be described later. In order to ascertain the configuration of the STR2, the CHN0 (1100) of the STR0 (1) becomes an initiator and issues to the STR2 a command to collect information via the CHF1 (1111). The CHN0 (1100) of the STR0 receives a response from the STR2 to the command via the CHF1 (1111) and recognizes from the information included in the response that the STR2 has a SATA disk pool and a low-cost, block type LU having a 15D + 1P configuration RAID 5 and with a large capacity of 2100 GB, whereby decides to manage the LU as a remote LU. Furthermore, due to the fact that the disk pool that the STR2 has is a large capacity, low-cost disk pool, the CHN0 (1100) of the STR0 determines the storage class of the disk pool as "Archive Storage." The CHN0 (1100) of the STR0 then assigns the number LU4 to the LU inside the STR2 and stores in a disk pool management table 131 of an SM 13 information concerning the LU, i.e., "Archive Storage" as the Storage Class #, "STR2" as the Storage Node #, "SATA pool" as the Disk Pool #, "LU4" as the LU #, "Remote Block" as the LU type, "RAID 5 15D + P" as the RAID Conf., and "2100 GB" as the Usable Capacity. When the CHN

cycle stage of the file, the Archive Storage class suitable for archiving is selected for files in "archive stage," or the old age, in its life cycle stage.

[0123]

Furthermore, other storage devices can be connected to the primary storage device, so that a storage hierarchy that takes advantage of differences in features of various storage devices can be constructed. Files can be migrated to LUs of the other storage devices, instead of migrating only within the primary storage device, according to the migration plan of each file; this further optimizes cost for storage devices compared to the situations in which a hierarchical storage control is realized using only one storage device.

[0124]

In addition, drives on disk devices that make up LUs whose storage class is "Archive Storage" can be halted to realize low power consumption and to extend the life of the disks.

[0125]

Moreover, due to the fact that even cheaper storage devices can be connected to the storage device STR1 according to the present embodiment, a storage hierarchy that is even more extensive can be set among a plurality of storage devices; by executing a hierarchical storage control using such a configuration, cost can be further optimized.

[0126]

Embodiment 3:





(1) Example of System Configuration (Fig. 14)

Next, with reference to Fig. 14, an example of the system configuration according to the third embodiment will be described. In the present embodiment, as in the second embodiment, a hierarchical storage control is executed among storage devices in a system in which another storage device STR2 (1b) is connected to a storage device STR0 (1) via a network. The third embodiment differs from the second embodiment in that while the network that connects the storage devices was the LAN 20 and file I/O interfaces were used between storage devices in the second embodiment, in the third embodiment, the network that connects the storage devices is a SAN 35, which is a dedicated network for connection between the storage devices, and a block I/O interface is used between storage devices.

[0127]

In Fig. 14, the storage device STR2 (1b) is a storage device with a small-scale configuration similar to the storage device STR1 (1a) in the second embodiment, but instead of the NAS controller NCTL0 of the storage device STR1 (1a) in the second embodiment, the storage device STR2 (1b) has SAN controllers FCTLx. FCTLx is provided with an FC controller 11012b to connect with the SAN 35, but it does not have the file access control CPU 11001a or its peripheral circuits as the STR1 does and does not perform file control. Otherwise, the storage device STR2 (1b) according to the present embodiment has a configuration similar to that of the storage

[0118]

The file storage management section 1100433 of the STR0 changes the link destination node name to STR1, the link destination FS name to LFS3, and the link destination filename to STR1-FILE00001 in the file storage management table 1100435 of the FILE00001 of the LFS2. The file storage management section 1100433 then opens all buffer management tables that can be referred to from the pointers registered in the file storage management table 1100435 of the FILE00001 and enters NULL in all the buffer management table entries of the file storage management table 1100435.

[0119]

The preceding transfers the substance of the data section of the file abc.doc from the FILE00001 in the LFS2 of the STR0 to STR1-FILE00001 in the RFS3 of the STR1.

[0120]

After this, when an access request is issued by any of the NAS hosts to access the file abc.doc, the CHN of the STR0 refers to the file storage management table of the abc.doc in the LFS0 and obtains its link destination node name, FS name and filename, and refers to the file storage management table of the FILE00001 in the LFS2 based on the identification information of the link destination obtained (i.e., STR0, LFS2, FILE00001). The CHN of the STR0 further obtains the link destination node name, the FS name and the filename from the file storage





management table of the FILE00001 of the LFS2 and issues to the NCTL of the STR1 an access request designating identification information of the link destination obtained (i.e., STR1, LFS3, STR1-FILE00001), which allows the CHN of the STR0 to reach STR1-FILE00001 of the RFS3 in the STR1 and access the data section of the abc.doc via the NCTL of the STR1.

[0121]

Due to the fact that a plurality of file storage management tables must be referred to in order to access a file that has been migrated to inside the LU3 of the STR1 according to the present embodiment, the access speed does suffer slightly. However, since files that are stored in the LU3 of the STR1 are files whose current life cycle stage is "archive stage" and therefore files that are rarely subjects of access requests, this poses no problem in practical terms. Even if an access request were to be issued from a host computer for data of a file in "archive stage," the data can be retrieved in real-time from its storage positions on disks where data is stored since it is stored on magnetic disks, even though it is a file that belongs to the Archive Storage class; unlike conventional situations in which such files are stored on tapes, neither enormous access time for tape control nor going back to transferring the data from the tape to a disk is required according to the present embodiment.

[0122]

As described above, according to the present embodiment, due to the fact that the storage positions of a file are determined according to the life

through a control of the file access control CPU 11001a; a file storage management table 1100435a and a file property information management table 1100438a are created within the file access control memory 11004a and information to be registered in each of the tables is set. The NCTL0 sends to the migration management section 110043A of the CHN0 the file handler assigned to the STR1-FILE00001, and the open processing is terminated.

[0112]

Next, as with the NAS host 0 (400) in the data write processing according to the first embodiment, the migration management section of the STR0 issues to the STR1 a write request containing the file handler obtained from the NCTL0 of the STR1 in the open processing, and requests to write actual data of abc.doc (i.e., data that is also actual data of FILE000001) as actual data of the file STR1-FILE00001.

[0113]

The file storage management section 1100433a of the STR1 secures buffer regions required to store the write data, determines storage positions on disks of the actual data of the file, and stores the write data received from the STR0 in the buffers.

[0114]

To determine the storage positions, the file storage management section 1100433a refers to the static property information of the file property information management table 1100438a. The file storage





management section 1100433a specifies the current life cycle stage of STR1-FILE00001 as "archive stage" due to the fact that the life cycle model of the file STR1-FILE00001 is "model 1" and to the fact that more than one year and one month have passed since the file was generated. Further, the file storage management section 1100344a specifies "Archive Storage" as the initial storage class as designated by the STR0.

[0115]

The file storage management section 1100433a sets the current life cycle stage as "archive stage" and the current storage class as "Archive Storage" in the life cycle information category of the dynamic property information of the file property information management table 1100438a. The file storage management section 1100433a further performs appropriate calculations for access information regarding the file STR1-FILE00001 and updates the information with the access information category of the file property information management table. Note that NULL is registered for all entries for link destinations in the file storage management table 1100435a of STR1-FILE00001.

[0116]

Next, under the control of the NCTL0, the data section of the file STR1-FILE00001 is stored at proper timing on disks that make up the LU3.

[0117]

This concludes the write processing in the STR1 and the processing returns to STR0.

110043A recognizes that the current life cycle stage has changed from the "reference stage" to the "archive stage" due to the fact that the life cycle model in the static property information for abc.doc indicates "model 1" and that one year, which is the period of the "reference stage," has already passed. Further, due to the fact that the migration plan is "plan 1," the migration management section 110043A recognizes that the file must be migrated from an LU whose storage class is "NearLine Storage" to an LU whose storage class is "Archive Storage".

[0106]

Next, the migration management section 110043A refers to the storage class management table 1100439, selects the LU3 that belongs to the "Archive Storage" class, and decides to transfer the file abc.doc to the LU3. The LU3 has attributes of "STR1 (i.e., the other storage device 1a)" as the storage node, "SATA disk pool" as the DiskPool #, and "Remote File" as the LU type.

[0107]

Next, the migration management section 110043A changes the current life cycle stage to "archive stage" and the current storage class to "Archive" in the dynamic property information of the file property information management table 1100438 for the abc.doc.

[0108]

Next, the migration management section 110043A defines a unique filename (in this case STR1-FILE00001) that is used to manage the file





abc.doc within the storage device STR0 (1).

[0109]

The migration management section 110043A behaves as if it were a NAS host, and issues to the STR1 an open request for the file STR1-FILE00001. This open processing is an open processing executed in order to store the file for the first time from the perspective of the STR1. For this reason, the STR0 includes in the open request the information that the STR0 has in the file property information management table as the static property information of the file abc.doc to send the information to the STR1. However, by changing only the initial storage class in the static property information to "Archive Storage" in the information and send thereof, the STR0 expressly designates to the STR1 to store the file STR1-FILE0001 in the Archive Storage class from the beginning.

[0110]

The NCTL0 of the STR1 receives the open request via a LAN controller 11002a, and a file access control CPU 11001a executes a file system program 110043a.

[0111]

When the file system program 110043a is executed, the open request received is specified in a manner similar to the first embodiment as an access request to access the local file system RFS3; the STR1-FILE00001 is registered in a filename management table 1100436a in a file access control memory 11004a and a file handler is assigned to the STR1-FILE00001

remote LU, i.e., as an LU that is in the other storage device STR1 (1a) but as one of the LUs that are managed by the primary storage device STR0 (1).

[0102]

The CHN0 assigns a number LU3 to the LU that the STR1 has and assigns a remote file system number RFS3 to the file system constructed within the LU. Due to the fact that the LU is in a large capacity, low-cost disk pool, the storage class of this LU is set as "Archive Storage." Through a control by a disk array control CPU 11008 of the CHN0, information regarding the LU3 existing in the STR1, such as the type of the abovementioned disk pool, the configuration of the LU, the LU number and the storage class, is stored in a disk pool management table 131 of an SM 13 of the storage device 1 (STR0). The CHN of the storage device 1 refers to the disk pool management table 131 by having a file access control CPU 11001 execute a file system program 110043, and can register information regarding the LU03 in a storage class management table 1100451 within a file access control memory by copying the information regarding the LU03 from the disk pool management table 131.

[0103]

As described in the first embodiment, let us assume that the NAS host 0 (400) has stored the file abc.doc in the LU0 of the STR0 via the CHN0 and that subsequently the file abc.doc was migrated to the LU2 of the STR0 through a control by the CHN0; in the present embodiment, only those parts that differ from the first embodiment in terms of the processing executed in





order to migrate the file abc.doc further to the LU3 in the other storage device STR1 are described.

[0104]

As described in the first embodiment, the file abc.doc's current life cycle stage is stored as "reference stage," its current storage class is stored as "NearLine Storage," and its data section is stored under the name FILE00001 in the LFS2 of the LU2 constructed in the SATA disk pool of the STR0, as shown in Figs. 11 and 12. The filename management table 1100436 in which the filename "abc.doc" is registered and the file storage management table 1100435 for the file abc.doc exist in the LFS0. In other words, information regarding the abc.doc is stored in the filename management table 1100436 for the LFS0 and in the file storage management table for the file abc.doc within the LFS0. In the meantime, the file property management information management table 1100438 is in both the LFS0 and the LFS2. Note that the data section of the file abc.doc has already been migrated to the LU2 in which is constructed the LFS2, which means that the data section of the abc.doc does not reside in the LU0, in which is constructed the LFS0.

[0105]

The migration management section 110043A of the STR0 refers to the file property information management table 1100438 of the abc.doc and compares the date created to the current date and time. Supposedly if one year has elapsed since the migration, the migration management section

[0097]

In Fig. 13, the storage device STR1 (1a) is the other storage device connected to the storage device STR0 (1) via a LAN 20; otherwise, the system configuration components are the same as in Fig. 1.

[0098]

In the STR1 (1a), an NCTL0 (1100a) and an NCTL1 (1101a) are NAS controllers, and a disk pool 0 (170a) is a disk pool connected to the NCTL0 and NCTL1.

[0099]

Instead of the SM I/F control circuit 11005 and the CM I/F control circuit 11006 in the configuration of the CHN 1100 according to the first embodiment shown in Fig. 4, the NAS controller NCTLx is provided with an FC controller 11010a for connecting with the disk pool 0 1700a. The NAS controller NCTLx also has a cache memory CM 14a within the NAS controller, as well as a data transfer circuit 11011a, which is a control circuit for the cache memory CM 14a. Further, the NAS controller NCTLx has the data transfer circuit 11011a, via which a NAS controller 1100a and NAS controller 1101b are connected to each other. Although details of the configuration of the NAS controller NCTL1 (1101a) are not shown in Fig. 13, the NAS controller 1101a has a configuration similar to that of the NAS controller 1100a. Note that components that are assigned with the same numbers as components of the CHN 1100 in the first embodiment have the similar configuration and the similar function as the corresponding





components of the CHN 1100.

[0100]

Let us assume that the STR1 is a storage device that is smaller and cheaper than the STR0. Also, as shown in Fig. 13, a CHN0 of the STR0 and the NCTL0 of the STR1 are connected via the LAN 20.

(2) Migration Processing of File to the Other Storage Device

The following is a description of the operation according to the present embodiment.

[0101]

The CHN0 (1100) of the storage device 1 (STR0) recognizes that the storage device 1a (STR1) of a different type is connected to the LAN 20. The different storage device can be recognized using a method based on information designated in advance by an administrator or a method based on whether or not there is a device that reacts to a broadcast of a command for recognition to network segments of the LAN 20. In order to ascertain the configuration of the STR1, the CHN0 of the STR0 becomes an initiator and issues to the STR1 a command to collect information. The response from the STR1 to the command includes the type of the disk pool and the configuration of LUs that the STR1 has; as a result, by referring to this response, the CHN0 can recognize that the STR1 processes the SATA disk pool 170a and that there is a low-cost file type LU having a 15D + 1P configuration RAID 5 and with a large capacity of 2100 GB in the disk pool 170a. The CHN0 of the STR0 then decides to manage the STR1's LU as a

access control memory 11004 as data of the FILE0001 to be written to the disks that make up the LU2, and the request processing section 1100432 writes the data to the storage regions in the buffers registered in the buffer management tables.

[0090]

The file storage management section 1100433 clears all the buffer management tables that can be referred to from pointers registered in the file storage management table 1100435 of the file abc.doc in the LFS0, and registers NULL in entries within these buffer management tables.

[0091]

The data of the FILE00001 stored in the buffers is stored at proper timing in the LU2 via the CM 14 of the storage device 1 through a procedure similar to the procedure that took place in the data write processing of the initial placement processing. This completes the migration processing.

[0092]

As described above, according to the present embodiment, files can be migrated to storage regions of an appropriate storage class by taking into consideration the life cycle stage of the file, based on the migration plan of the file.

[0093]

According to the present embodiment, LUs for storing files can be selected based on a concept of storage classes, and LUs for storing files can be changed, without being dependent on host computers or applications





executed on the host computers. As a result, a storage device with storage hierarchy, i.e., a plurality of storage regions with varying properties, having high cost effectiveness can be realized without being dependent on the host computers.

[0094]

Further, due to the fact that data is migrated on a per-file basis, same files can be accessed from a plurality of host computers using a file I/O interface, even after the files are migrated.

[0095]

Moreover, a file-based hierarchy storage control can be executed based on static properties of the file, such as the file type, the type of application that generated the file, the intent (policy) of the file generator, and on dynamic properties of the file, such as changes in the life cycle stage, value and access property of the file.

[0096]

Embodiment 2:

(1) Example of System Configuration (Fig. 13)

Next, referring to Fig. 13, an example of the system configuration of the second embodiment will be described. In the present embodiment, a hierarchical storage control is executed between storage devices in a system in which a storage device 1 (hereinafter called "STR0") described in the first embodiment and another storage device 1a (hereinafter called "STR1") are connected via a network.

information management table 1100438.

[0084]

The migration management section 110043A then defines a unique filename (in this case FILE00001) that is used to manage the file abc.doc within the storage device STR0 (1).

[0085]

The file open processing section 1100431 refers to the filename management table 1100436 of the LFS2 (60) and confirms whether the filename FILE00001 has not yet been registered in the filename management table 1100436; if it has not been registered, the file open processing section 1100431 registers the filename FILE00001 in the filename management table 1100436 and assigns a file handler to the filename FILE00001.

[0086]

Next, the file storage management section 1100433 generates the file storage management table 1100435 and the file property information management table 1100438 by corresponding to the file handler assigned to the file FILE00001. Contents identical to the contents registered in the file property information management table of abc.doc are stored in the file property information management table 1100438 generated. The file storage management section 1100433 then writes in the LU, which stores FILE00001, the file storage management table and the file property information management table of FILE00001.





[0087]

Next, the file storage management section 1100433 secures buffer regions required to store the data of FILE00001 and determines the storage regions (or the storage positions) within the LU2 for storing the file. Using a method similar to the method used in the data write processing, the file storage management section 1100433 generates the buffer management tables 1100438 to register the storage positions determined, and stores in the buffer management table entry of the file storage management table 1100435 pointers to the buffer management tables 1100438 generated. Note that all entries for the link destinations in the file storage management table 1100435 of the filename FILE00001 stored in the LFS2 are NULL.

[0088]

Moreover, as indicated in Fig. 12, the file storage management section 1100433 changes the link destination node name to STR0, the link destination FS name to LFS2, and the link destination filename to FILE00001 in the file storage management table 1100435 of abc.doc in the LFS0.

[0089]

Next, the request processing section 1100432 reads data of the abc.doc from disks that make up the LU0 to buffers within the file access control memory 11004. The abovementioned file storage management section 1100433 determines the data read to the buffers within the file

[0076]

Next, under the control of the DKA 120 that controls disks that make up the LU0, the write data is stored on appropriate disks at proper timing.

[0077]

As described above, files can be initially placed in storage regions that belong to the appropriate storage class based on the static property information of the file.

(16) File Migration Processing (Fig. 12)

Next, a migration processing of a file will be described.

[0078]

The migration management section 110043A of the file system program 110043 is activated by the file access control CPU at a preset timing.

[0079]

The migration management section 110043A then, for the local file system set in advance as the subject of the migration processing, refers to the file property management table of a file included in the file system of the local file system, and checks whether the file that is the subject of migration exists. The following is a detailed description of a situation in which the file abc.doc is the subject of the migration processing.

[0080]

The migration management section 110043A refers to the file





property information management table 1100438 of the file abc.doc and compares the date created to the current date and time. Supposedly, if one month has elapsed since the date created, the migration management section 110043A recognizes that the current life cycle stage has changed from the "update stage" to the "reference stage" due to the fact that the life cycle model in the static property information indicates "model 1" and that one month, which is the period of the "update stage," has already passed.

[0081]

Further, due to the fact that the migration plan is "plan 1," the migration management section 110043A recognizes that the file must be migrated from the LU whose storage class is the "OnLine Storage (Premium)" to an LU whose storage class is the "NearLine Storage."

[0082]

The migration management section 110043A then refers to the storage class management table 1100439 and decides to transfer the file to an LU whose storage class belongs to the "NearLine Storage" and that is designated by "STR0 (i.e., the primary storage device 1)" as the storage node, "SATA disk pool 1710" as the DiskPool #, and "LU2 (i.e., a local file system LFS2)" as the LU #.

[0083]

Next, the migration management section 110043A changes the current life cycle stage to "reference stage" and the current storage class to "NearLine Storage" in the dynamic property information of the file property

access taking place within one month of the file generation since it is an access request occurring in an initial file placement, the file storage management section 1100433 specifies the current life cycle stage of the file abc.doc as "growth stage." Further, since the initial storage class is "undesignated" and the asset value type is "important," the file storage management section selects "OnLine Storage (Premium)" as the storage class of the storage pool in which to store the file abc.doc.

[0073]

Next, the file storage management section 1100433 refers to the storage class management table 1100439 and decides to store the file in an LU whose storage class is "OnLine Storage (Premium)" and that is specified by "STR0 (i.e., the primary storage device 1)" as the storage node, "FC disk pool 1700" as the DiskPool #, and "LU0 (i.e., the local file system LFS0)" as the LU #. The file storage management section 1100433 divides the data within the file into one or more logical blocks based on an appropriate algorithm, determines a storage address in the LU0 for each of the logical blocks, generates buffer management tables 1100437 to register the storage addresses determined, and stores in the buffer management table entry of the file storage management table 1100435 pointers to the buffer management tables generated. Furthermore, the file storage management section 1100433 stores information in the remaining entries of the file storage management table 1100435. Note that, in the present embodiment, all entries for the link destinations in the file storage management table are





NULL.

[0074]

The file storage management section 1100433 then sets the current life cycle stage as "update stage" and the current storage class as "OnLine Storage (Premium)" for the life cycle information category of the dynamic property information of the file property information management table 1100438. Moreover, the file storage management section 1100433 performs appropriate calculations for information included in the access information category of the dynamic property information, and then registers the results into the file property information management table 1100438.

[0075]

The request processing section 1100432 executes a processing according to the write request received; and the LAN controller driver program 110041, the TCP/IP program 110042, and the network file system program 110044 are executed by the file access control CPU 11001; as a result, the write data is transferred from the NAS host 0 (400) to the CHN0 (1100) and temporarily stored in the buffer of the file access control memory 11004. Next, the inter-CPU communications driver program 110046 is executed by the file access control CPU 11001, and this causes the write request to be transferred to the disk array control CPU at proper timing. Upon receiving the write request, the disk array control CPU 11008 caches the write data temporarily in the CM 14 and sends a reply of completion with regard to the write request from the NAS host 0 (400).

Next, the file storage management section 1100433 creates the file storage management table 1100435 to correspond to the file handler assigned to the file abc.doc.

[0066]

Next, the file storage management section 1100433 generates the file property information management table 1100438, correlates it to the file storage management table 1100435 (i.e., a pointer to the file property information management table is stored in the file storage management table)n and then stores in the file property information management table 1100438 the static property information of the file property information obtained from the NAS host 0, as well as the date created and owner of the file. Next, the file storage management table and the file property information management table are written to the LU in which is constructed the file system the file belongs to.

[0067]

Next, the CHN0 (1100) returns the file handler to the NAS host 0 and the open processing is terminated.

(15) Initial File Placement: A Data Write Processing

Next, a description will be made as to a data write processing executed in the initial placement processing of a file.

[0068]

Using the file handler obtained in the open processing, the NAS host 0 (400) issues to the CHN0 (1100) a write request in order to store data of





the file abc.doc in the storage device 1.

[0069]

When the write request is received by the CHN0 (1100), the file access control CPU 11001 executes the file system program 110043 and uses a method similar to the method used in the open processing in order to specify that the write request is an access request to access the local file system LFS0 (60).

[0070]

The request processing section 1100432 of the file system program 110043 interprets the access request as a write request based on the information included in the access request received, and uses the file handler designated in the write request to obtain the file storage management table 1100435 of the file that corresponds to the file handler.

[0071]

Next, the file storage management section 1100433 secures buffers required to store the data and determines the storage positions on disks for the file.

[0072]

To determine the storage positions, the file storage management section 1100433 refers to the static property information of the file property information management table 1100438. In this case, due to the fact that the life cycle model of the file abc.doc, which is the subject of the write request, is "model 1," and to the fact that the write request received is an

The life cycle information category includes information related to the life cycle of a file. In the life cycle information category, a current life cycle stage indicates the current positioning of a file within its life cycle, i.e., the update stage, the reference stage, or the archive stage. A current storage class indicates the storage class of a storage pool where the LU, which currently stores the file, is set.

[0060]

Fig. 11 indicates one example of the file property information, but various other types of property information can be defined and stored in the file property information management table 1100438. Furthermore, an embodiment may use only a part of the property information as necessary.

(14) Initial File Placement: A File Open Processing

Next, a description will be made as to a file open processing that takes place in the initial placement processing to store a file in a storage device for the first time.

[0061]

Let us assume that the NAS host 0 (400) generated a file abc.doc.

[0062]

The NAS host 0 (400) issues to the CHN0 (1100) an open request for the file abc.doc. The open request includes a filename as identification information to identify the file. Since this open processing is executed to newly store the file, the NAS host 0 (400) sends to the CHN0 (1100) the following information included in the file information category and the





policy category as the static property information of the file property information, along with the open request. The information sent includes a file type "document," an application "XYZ Word" that generated the file , and an access identifier "-rw-rw-rw-" as information included in the file information category, as well as an initial storage class "undesignated," an asset value type "important," the life cycle model "model 1," and the migration plan "plan 1" as information included in the policy category.

[0063]

The CHN0 (1100) receives the open request of the file from the NAS host via the LAN controller 11002, and the file access control CPU 11001 executes the file system program 110043.

[0064]

When the file system program 110043 is executed, the open request received is specified through a control by the file access control CPU 11001 as an access request to access the local file system LFS0 (60) based on the directory information of the filename. The file open processing section 1100431 refers to the filename management table 1100436 of the LFS0 (60) and searches for abc.doc. Since it is determined as a result that abc.doc is a file that does not yet exist in the filename management table 1100436 and is to be stored for the first time, the file open processing section 1100431 registers abc.doc in the filename management table and assigns a file handler to abc.doc.

[0065]

One example is a method of defining the "growth stage," or the "update stage," in which there are frequent updates, as one month; the "mature stage," or the "reference stage" as one year; and the "old age," or the "archive stage," as thereafter. Hereinafter this definition is called a "model 1" and is used in the following description. By varying the time interval of the life cycle model or by defining stages with finer resolution, various life cycle models can be defined and one certain life cycle model from among a plurality of life cycle models can be selected for use. A specific life cycle model can be provided to be applied to a certain type of files, or life cycle models can be applied on a per-application basis to files created by the application. Names of the life cycle stages may be expressed in terms of "growth stage," "mature stage," and "old age" that correspond to the life of a person, or in terms of "update stage," "reference stage," and "archive stage" based on file behavior. In the present embodiment, the latter expressions will be used in order to more clearly indicate the behavior of files.

[0057]

The migration plan defines to which storage class of LU a file is transferred according to the file's life cycle stage. One example is a method for storing "update stage" files in OnLine Storage class LUs, "reference stage" files in the NearLine Storage class LUs, and "archive stage" files in Archive Storage class LUs. Hereinafter, this definition is called a "plan 1" and is used in the following description. In addition to this plan, various plans can be defined, such as a plan that defines "update stage" files to be





stored in OnLine Storage (Premium) class LUs, and "reference stage" files in OnLine Storage (Normal) class LUs, while "archive stage" files remain in the NearLine Storage class LUs, and one plan from among a plurality of plans can be selected for use. Furthermore, a specific migration plan may be applied to a specific type of files, or migration plans may be provided to be applied on a per-application basis to files created by the application.

(13) Dynamic Property Information

for the dynamic property information, there is an access information category and a life cycle information category.

[0058]

The access information category includes access statistical information for a file. In the access information category, a time stamp indicates the date and time a file was last read and written, or the date and time the file storage management table of the file was last updated. An access count indicates the total number of accesses to the file. A read count and a write count indicate the number of reads and the number of writes, respectively, to and from the file. A read size and a write size indicate the average value of the data transfer size when reading and writing, respectively, to and from the file. A read sequential count and a write sequential count indicate the number of times there is address continuity, i.e., sequentiality, between two of multiple consecutive accesses in reading and writing respectively.

[0059]

[0052]

The file information category includes basic information of a file. In the file information category, a file type indicates the type of the file, such as a text file, document file, picture file, moving picture file or a voice file. An application indicates the application that generated the file. A date created indicates the date the file was first generated. The time at which the file was generated can be registered as the date created, in addition to the date the file was created. An owner indicates the name of the user who created the file. An access identifier indicates a range of access authorization for the file.

[0053]

The policy category is information that is set by the user or the application that created the file, and is information that is designated by the user or the application with regard to file storage conditions and the like. An initial storage class is information that indicates the storage class of the LU in which the file is to be stored when the file is stored in a storage device for the first time. An asset value type indicates the asset value of the file. A life cycle model indicates the model applicable to the file from among the life cycle models defined in advance. A migration plan indicates the plan applicable to the file from among the plans concerning file migration (hereinafter called "migration") defined in advance.

[0054]

The asset value is an attribute that designates the importance or





value attached to the file. An attribute of "extra important," "important" or "regular" etc, for example, can be designated as an asset value. The asset value can be used as a supplemental standard for selecting a storage class, i.e., storing files with an attribute of "important" or higher in LUs that belong to the storage class with Premium attribute, or as a standard for selecting a storage class when no life cycle models are designated, for example.

[0055]

In the description of the present embodiment below, it will be assumed that files that are "important" or higher are stored in LUs that belong to the storage class of "Premium" class. Needless to say, the present invention is not restricted to such an assumption, and different standards can be used to select storage classes of LUs for storing files.

[0056]

The life cycle stages have been named by drawing analogy with life cycle stages of humans to describe how the usage status of a file changes over time, i.e., the period in which data is created is the birth, the period in which the data is updated and/or used is the growth stage, the period in which the data is rarely updated and is mainly referred to is the mature stage, and the period in which the data is no longer used and is archived is the old age. A life cycle model defines the life cycle a file experiences. The most general method of defining a life cycle includes a method of defining the stages based on the time that has elapsed since a file was generated.

management table 1100435 and the buffer management table 1100437. The file storage management table is provided in the file access control memory for each file and is a table that manages file storage addresses. The file storage management table can be referred to by designating a file handler that indicates a file.

[0049]

A file property information management table column stores a pointer for referring to the file property information management table 1100438 for the corresponding file. A size indicates the size of the file in units of bytes. A number of blocks represent the number of logical blocks used in managing the file by dividing the file into blocks called logical blocks. Each logical block that stores the file also stores a pointer to the buffer management table 1100437 that corresponds to the logical block.

[0050]

There is one buffer management table 1100437 for each logical block, and each buffer management table 1100437 contains the following. A hash link column stores a link pointer to a hash table for quickly determining whether a buffer is valid. A queue link column stores a link pointer for forming a queue. A flag column stores a flag that indicates the status of the corresponding buffer, i.e. whether valid data is stored in the buffer, whether the buffer is being used, whether the content of the buffer is unreflected on the disk and so on. An equipment number column stores an identifier of the storage device and an identifier of the LU in which the





corresponding logical block is stored. A block number column stores a disk address number that indicates the storage position of the logical block within the storage device indicated by the equipment number. A number of bytes column stores the number of bytes of valid data stored in the logical block. A buffer size column stores the size of the buffer in units of bytes. A buffer pointer column stores a pointer to the corresponding physical buffer memory.

[0051]

The file storage management table is stored in the LU that stores the corresponding file and is used by being read to the memory when necessary for use.

(11) File Property Information Management Table (Fig. 11)

Fig. 11 is a diagram indicating an example of the file property information management table 1100438 stored in the file access control memory 11004. The file property information management table stores the static property information and dynamic property information. The static property information is determined when a file is configured and carries over thereafter. Needless to say, although the static property information can be intentionally altered, it otherwise remains unaltered. The dynamic property information changes over time after a file is created.

(12) Static Property Information

For the static property information, there is a file information category and a policy category.

the structure of the corresponding disk array, such as data record and parity record number within a parity group, is stored. In a Usable Capacity column (1100451g) and a Used Capacity column (1100451h), information that indicates the total storage capacity of the LU and the storage capacity being used, is stored.

[0046]

A storage class is a hierarchical attribute of the storage region, which is provided for each usage of data storage; according to the present embodiment, three attributes of OnLine Storage, NearLine Storage and Archive Storage are defined. In addition, sub-attributes of Premium and Normal are defined for the OnLine Storage. The OnLine Storage is an attribute set for LUs suitable for storing data of files that are frequently accessed, such as files being accessed online and files being generated. Particularly, Premium indicates an attribute set for LUs suitable for storing data requiring fast response. The NearLine Storage is an attribute set for LUs suitable for storing data of files that are not frequently used but are occasionally accessed. The Archive Storage is an attribute set for LUs suitable for storing data of files that are hardly ever accessed and are maintained for long-term storage.

[0047]

Fig. 8 indicates that there are the LU0 (50) of the OnLine Storage (Premium) class and the LU1 (51) of the OnLine Storage (Normal) class in the FC disk pool 170 of the storage device 1 (called "STR0"). Further, in





the SATA disk pool 171 within the same storage device 1 (STR0) exists the LU2 (52) of the NearLine Storage class. Moreover, in a different storage device (STR1) exists an LU3 (53) of the Archive Storage class in a SATA disk pool. An example of constructing disk pools in different storage devices is described later.

(9) Filename Management Table (Fig. 9)

Fig. 9 shows an example of the filename management table 1100436 that is stored in the file access control memory 11004. The filename management table 1100436 is a table prepared for each file system, where filenames and file handlers are stored in a tree structure for easy searchability. When a file is accessed by the NAS hosts 40x, the filename is included in an access request received by the CHN from the NAS host. The CHN uses this filename to search the filename management table 11004 and obtains the file handler that corresponds to the filename, which enables the CHN 110x to refer to the file storage management table 1100435 that corresponds to this file handler.

[0048]

The filename management table is stored in the LU in which the file system that corresponds to the filename management table is constructed, and is read the memory when necessary and used by the file access control CPU.

(10) File Storage Management Table (Fig. 10)

Fig. 10 is a diagram indicating an example of the file storage

[0041]

LU2 (52) is established in the SATA disk pool 171. The LU2 (52) consists of nine SATA disks, DK100, DK101, DK102, DK103, DK104, DK110, DK111, DK112 and DK113, where these nine SATA disks make up an 8D + 1P configuration RAID 5.

[0042]

When the capacity of each disk is 140 GB, the LU0 (51) has 140 GB, the LU1 (52) has 560 GB, and the LU2 (53) has 1120 GB in usable storage capacity.

[0043]

Here, independent local file systems LFS0 (60), LFS1 (61) and LFS2 (62) are established and constructed for the LUs respectively.

[0044]

(8) Storage Class Management Table (Fig. 8)

Fig. 8 indicates an example of the configuration of a storage class management table 1100451 stored in the file access control memory 11004 of each CHN 110x. The storage class management table 1100451 is created by the file access control CPU 11001's executing the file system program 110043 and by referring to information stored in the disk pool management table 131 of the SM 13.

[0045]

Although the disk pool management table 131 is not shown, the disk pool management table 131 is stored in the SM 13 and contains information





similar to the information in the storage class management table 1100451 for all CHs. In other words, of the information in the disk pool management table 131, the storage class management table 1100451 stored in the file access control memory 11004 of a certain CHN 110x is, for the information regarding LUs used by the CHN 110x, rearranged with the storage class as a key.

The following is a description of the configuration of the storage class management table. A storage class column (1100451a) stores information indicating storage class. A storage node # column (1100451b) stores an identification number (called a "storage node number") of the storage device that makes up the storage class. A disk pool # column (1100451c) stores a disk pool number that makes up the storage class. An LU # column (1100451d) stores an LU number set for the disk pool. In an LU type column (1100451e), information, which indicates whether the corresponding LU is set internally (Local) or externally (Remote) to the given storage device and whether a file system is set in the LU, is stored. In other words, if the LU exists within the storage device, "Local" is registered in the LU type column, while "Remote" is registered in the LU type column if the LU exists in a different storage device; if a file system is constructed in the LU, "File" is registered in the LU type column, while "Block" is registered in the LU type column if no file systems are constructed in the LU. In a RAIDConf. column (1100451f), information, which indicates the RAID level of the disk array that makes up the LU and

9) a migration management section 110043A used when executing a processing to migrate files between LUs; and

10) a storage class management table 1100439 that registers for each LU made up in the storage pool a storage class, described later, and identification information of the storage device in which an LU resides.

[0039]

(6) Configuration of Disk Array Control Memory (Fig. 6)

Fig. 6 is a diagram of an example of programs stored in the disk array control memory 11009. An operating system program 110090 is used for managing programs as a whole and for controlling input/output. A disk array control program 110091 is used for constructing LUs within the disk pools 17x and for processing access requests from the file access control CPU 11001. A disk pool management program 110092 is used for managing the configuration of the disk pools 17x by using information in the disk pool management table 131 stored in the SM 13. An inter-CPU communications driver program 110093 is used for controlling the inter-CPU communications circuit 11007 for performing communications between the file access control CPU 11001 and the disk array control CPU 11008. A cache control program 110094 is used for managing data stored in the CM 14 and for controlling cache hit/miss judgments and the like. A DKA communications driver program 110095 is used, when accessing an LU, in order to communicate with the DKAs 12x that control the disks 170x and 171x that make up the LU.





(7) Configuration of Disk Pools (Fig. 7)

Fig. 7 is a diagram indicating an example of the configuration of the disk pools.

[0040]

In the FC disk pool 170 are set two LUs, LU0 (51) and LU1 (52). The LU0 (51) comprises two FC disks, DK000 and DK010, where the DK000 and the DK010 make up RAID 1. The LU1 (52) consists of five FC disks, DK001, DK002, DK003, DK004, and DK005, where these five FC disks make up a 4D + 1P configuration RAID 5. The RAID 1 and the RAID 5 refer to data placement methods in a disk array respectively and are discussed in detail in "A Case for Redundant Arrays of Inexpensive Disks (RAID)" by D. Patterson and two others, ACM SIGMOD Conference Proceedings, 1988, pp. 109-116. In the LU0 with the RAID 1 configuration, the two FC disks, DK000 and DK010, have a mirror relationship with each other. In the meantime, the LU1 having the RAID 5 configuration consists of one or more disks that store data stripes, which store data of files accessed from host computers, and one or more disks that store parity stripes, which are used to retrieve data stored in the data stripes. The LU1 has the 4D + 1P configuration RAID 5, which indicates that the "4D + 1P" is a RAID 5 consisting of four data stripes and one parity stripe. Similar representations will be used hereinafter to indicate the number of data stripes and the number of parity stripes in LUs having the RAID 5 configuration.

the inter-CPU communications circuit 11007, are replaced by a Fiber Channel controller.

[0037]

(5) Example of Programs Stored in File Access Control Memory (Fig. 5)

Fig. 5 is a diagram indicating an example of programs and control data stored in the file access control memory 11004 contained in the CHN 110x. An operating system program 110040 is used for the management of programs as a whole and for input/output control. A LAN controller driver program 110041 is used for the control of the LAN controller 11002. A TCP/IP program 110042 is used for the control of TCP/IP, which is the communications protocol for the LAN. A file system program 110043 is used for managing files stored in the storage device. A network file system program 110044 is used for controlling NFS and/or CIFS, which are protocols for providing files stored in the storage device to the NAS hosts 40x. A volume control program 110045 is used for controlling the configuration of each logical volume by combining a plurality of logical disk units (hereinafter called "LU"), each of which is a unit of storage region set within the disk pools 17x. An inter-CPU communications driver program 110046 is used for controlling the inter-CPU communications circuit 11007 for performing communications between the file access control CPU 11001 and the disk array control CPU 11008.

[0038]

The file system program 110043 comprises:





1) a file open processing section 1100431 for executing a file open processing when using a file;

2) a request processing section 1100432 for executing processing according to an access request when a file access request is received;

3) a file storage management section 1100433 for dividing each file into blocks, determining the storage position on a disk for each block, and managing the storage position of each block;

4) a buffer management section 1100434 for managing correlation between each block and a buffer configured in the memory;

5) a file storage management table 1100435, which is used for managing addresses of storage regions on disks that store blocks that make up each file;

6) a filename management table 1100436 for managing filenames of opened files and file handlers used to access the file storage management table 1100435 of the opened file;

7) a buffer management table 1100437 for managing buffer addresses indicating storage regions within buffers corresponding to blocks that make up a file;

8) a file property information management table 1100438 for storing file static properties, such as the file type, the application that generated the file, the intent of the file generator, and file dynamic properties, such as the value of the file that varies according to the file's life cycle stage and the file's access properties;

connector of the DKC unit 19. An interface connector 2001 is Ethernet ®-compatible and can be connected to Ethernet ®.

[0032]

According to the present embodiment, due to the fact that the shape of the connector on the adapter boards is uniform regardless of the type of the adapter boards as described earlier, the adapter boards with built-in CHNs 110x and the adapter boards with built-in CHFs 111x have connectors of the same shape. On the adapter boards with the built-in CHFs 111x, the interface connector 2001 is Fiber Channel-compatible and configured to be connected to Fiber Channel.

[0033]

(4) Example of Configuration of NAS Board (or CHN) (Fig. 4)

Fig. 4 is a diagram indicating an example of the configuration of the CHN 110x. A file access control CPU 11001 is a processor for controlling file access. A LAN controller 11002 is connected to the LAN via the interface connector 2001 and controls sending and receiving of data to and from the LAN. A file access control memory 11004 is connected to the file access control CPU 11001. The file access control memory 11004 stores programs executed by the file access control CPU 11001 and control data.

[0034]

A disk array control CPU 11008 is a processor for controlling a disk array. The disk array here refers to a storage device consisting of a plurality of disks. Disk arrays in which at least one of a plurality of disks





stores redundant data to provide fault tolerance are called RAIDs. RAIDs are described later. A disk array control memory 11009 is connected to the disk array control CPU 11008 and stores programs executed by the disk array control CPU 11009 and control data. An SM I/F control circuit 11005 is a circuit for controlling access from the CHNs 110x to the SM 13. A CM I/F control circuit 11006 is a circuit for controlling access from the CHNs 110x to the CM 14. An inter-CPU communications circuit 11007 is a communications circuit used when the file access control CPU 11001 communicates with the disk array control CPU 11008 in order to access disks.

[0035]

The present embodiment indicates an example of an asymmetrical multiprocessor configuration in which two processors, the file access control CPU 11001 and the disk array control CPU 11008, are mounted on each CHN; however, each CHN can be implemented by configuring same to execute the file access control and the disk array control by means of a single processor, or implemented as a symmetrical multiprocessor configuration in which two or more processors execute the file access control and the disk array control as equals.

[0036]

The configuration of each CHF111x is the configuration, in which the components shown in top half of Fig. 4, namely the LAN controller 11002, the file access control CPU 11001, the file access control memory 11004 and

However, an IP network can be used as the SAN, such that iSCSI, by which SCSI commands according to SCSI protocol are encapsulated with IP packets for sending and receiving, is used among equipment connected to the SAN. The SAN 35 according to the present embodiment is a dedicated SAN for connecting the storage device 1 and no SAN hosts are connected to the SAN 35.

[0026]

In the storage device 1, all CHs can access the CM 14, the SM 13, any DKAs 12x and any disks 17x, via the CMC 16 or the SMC 15.

[0027]

The storage device 1 shown in Fig. 1 has both the SAN interfaces (CHFs 111x) for connecting to the SAN hosts 50x and the NAS interfaces (CHNs 110x) for connecting to the NAS hosts 40x, but the present embodiment can be implemented even if the storage device 1 has only the NAS interfaces.

[0028]

(2) Example of Exterior Appearance of Storage Device (Fig. 2)

Fig. 2 is a diagram indicating an example of the exterior appearance of the storage device 1.

[0029]

A DKC unit 19 stores the CHNs 110x, the CHFs 111x, the DKAs 12x, the SM 13 and the CM 14, which are components of the DKC 11. The SM 13 actually comprises a plurality of controller boards 13x. The CM 14 also





comprises a plurality of cache boards 14x. Users of the storage device 1 can increase or decrease the number of such boards in order to configure the storage device 1 with the CM 14 and the SM 13 having the desired storage capacity. Disk units (hereinafter called "DKU") 180 and DKUs 181 store the disk pool 170 and the disk pool 171, respectively.

[0030]

Adapter boards built-in with the CHNs 110x, the CHFs 111x, the DKAs 12x, the controller boards 13x and the cache boards 14x are stored in slots 190. According to the present embodiment, the shape of the slots 190, the size of the adapter boards and the shape of connectors are made uniform regardless of the type of adapter boards or the type of interfaces, which maintains compatibility among various types of boards. As a result, in the DKC unit 19, any adapter board can be mounted into any slot 190 regardless of the type of the adapter board or the type of the interface. Furthermore, users of the storage device 1 can freely select the number of adapter boards for the CHNs 110x and the CHFs 111x in order to mount the selected number of the CHNs and the CHFs into the slots of the DKC unit 19.

[0031]

(3) Example of Exterior Configuration of Adapter Board (hereinafter called "NAS board") with the CHN 110x Built-in (Fig. 3)

Fig. 3 is a diagram indicating an example of the exterior configuration of a NAS board. A connector 11007 is connected to a

controller (hereinafter called "DKC") 11 and a plurality of magnetic disk devices (hereinafter simply called "disks") 170x and 171x. In the present embodiment, the storage device 1 is provided with two types of disks 170x and 171x. 170x are Fiber Channel (hereinafter called "FC") disks with FC-type interface, while 171x are serial AT attached (hereinafter called "SATA") disks with SATA-type interface. A plurality of FC disks 170x makes up an FC disk pool 0 (170), while a plurality of SATA disks 171x makes up a SATA disk pool 1 (171). The disk pools will be described in detail later.

[0019]

Next, the configuration of the DKC 11 of the storage device 1 will be described. The DKC 11 comprises one or more NAS channel adapters 110x, one or more Fiber Channel adapters 111x, a plurality of disk adapters 12x, a shared memory 13 (hereinafter called "SM"), a shared memory controller 15 (hereinafter called "SMC"), a cache memory 14 (hereinafter called "CM"), and a cache memory controller 16 (hereinafter called "CMC").

[0020]

The NAS channel adapters (hereinafter called "CHN") 110x are interface control devices connected by file I/O interfaces to computers 40x (hereinafter called "NAS hosts"), which are connected to a local area network (hereinafter called "LAN") 20 or a LAN 21.

[0021]

The Fiber Channel adapters (hereinafter called "CHF") 111x are





interface control devices connected by block I/O interfaces to computers (hereinafter called "SAN hosts") 50x, which are connected to a storage area network (hereinafter called "SAN") 30. Hereinafter, CHN and CHF are collectively called channel adapters (hereinafter called "CH").

[0022]

The disks 17x are connected to the disk adapters 12x. Each disk adapter (hereinafter called "DKA") 12x controls input and output to and from one or more disks 17x connected to itself.

[0023]

The SMC 15 is connected to the CHN 110x, the CHF111x, the DKA 12x and the SM 13. The SMC 15 controls data transfer among the CHN 110x, the CHF111x, the DKA 12x and the SM 13. The CMC 16 is connected to the CHN 110x, the CHF111x, the DKA 12x and the CM 14. The CMC 16 controls data transfer among the CHN 110x, the CHF111x, the DKA 12x and the CM 14.

[0024]

The SM 13 stores a disk pool management table 131. The disk pool management table 131 is information that is used to manage the configuration of the disk pools.

[0025]

The LANs 20 and 21 connect the CHNs 110x to the NAS hosts 40x. Generally, Ethernet ® is used for LAN. The SAN 30 connects the CHFs 111x to the SAN hosts 50x. Generally, Fiber Channel is used for SAN.

magnetic disks, it is difficult to use magnetic tapes as storage device for online access.

According to Patent Document 2, a hierarchical storage control is executed by taking advantage of the difference in price and performance resulting from different RAID configurations of magnetic disks; however, since the price difference results only from the difference in the degree of redundancy in RAID configurations, only the cost reduction equivalent only to the difference in the degree of redundancy can be hoped for.

[0012]

The object of the present invention is to provide a control method or a storage device that can execute a hierarchical storage control for file storage positions without being dependent on the OS or applications executed on a host computer.

[0013]

Another object of the present invention is to provide a hierarchical storage control method for a plurality of computers to be able to share files, or to provide a storage device that executes such a hierarchical storage control.

[0014]

Another object of the invention is to provide a control method or a storage device that can execute a hierarchical storage control according to file properties.

[0015]





Yet another object of the present invention is to provide a hierarchical storage control method with high cost reduction effect, or to provide a storage device that executes such a hierarchical storage control.

[0016]

[Means for Solving the Problem]

A storage device comprises a plurality of storage regions having different properties, an interface control device that accepts from one or more computers access requests having file identification information, and an interface control device for accessing storage regions that store data of the file designated by the identification information, wherein the interface control device controls to store data of the file in one of the plurality of storage regions according to the file properties.

[0017]

[Embodiment of the Invention]

Embodiments of the present invention will be described bellow. However the following embodiments do not limit the present invention.

[Embodiment 1]

(1) Example of System Configuration (Fig. 1)

Fig. 1 is a diagram indicating an example of a computer system comprising a storage device 1, to which the present invention is applied. In the following, x may be any integer.

[0018]

The storage device 1 is a disk array system comprising a disk

processing speed and/or storage capacity, and to store the data in the storage region selected. However, when the system configuration is altered, such as when an old computer is replaced by a new computer, maintaining the system can be difficult due to such reasons as the system configuration of the new computer not being able to take over the software's control information.

[0007]

According to Patent Document 2, although a hierarchical storage control is implemented on a per-logical storage device basis, a technology for the storage device to recognize a data structure of data stored in the logical storage device or a technology for executing exclusive control are not disclosed. As a result, it would be difficult for a plurality of computers to share the same logical storage devices, and integrating storage devices used by a plurality of computers in order to reduce the management cost of the computer system would require imposing certain limitations on the configuration of the computer system, such as allocating a logical storage device for each computer.

[0008]

The second problem is that optimal placement of data according to the life cycle or type of data is difficult.

[0009]

According to the conventional technology, data that had high access frequency in the past is assumed to have high access frequency in the future





as well, and the storage regions in which the data is stored are determined based on statistical information regarding data access frequency and on used capacity of storage regions that can be accessed at high-speed. The processing efficiency can be improved by increasing the probability with which data with high access frequency can reside in a storage device that can be accessed at high-speed. However, there are no technologies disclosed for determining storage regions in which to store data by taking into consideration differences in data properties that are dependent on the data's life cycle stage, i.e., the time elapsed since the corresponding file was generated, the type of application that generates and uses the data, and the type of data itself.

[0010]

The third problem is that the effect of the hierarchical storage control is small.

[0011]

According to Patent Document 1, a hierarchical storage control is executed by taking advantage of the difference in capacity and price between magnetic tapes and magnetic disks; however, the difference in capacity and price between magnetic tapes and magnetic disks have been growing smaller in recent years; consequently, the effect of cost optimization and cost reduction through the use of hierarchical storage control has also been growing smaller. Furthermore, due to the fact that the access speed to magnetic tapes is extremely slow compared to the access speed to

[Detailed Description of the Invention]

[0001]

[Technical Field to which the Invention Pertains]

The present invention relates to a storage device used in a computer system.

[0002]

[Prior Art]

A system, in which is disclosed in Patent Document 1 a high-speed storage device and a low-speed storage device are connected to a computer, called a hierarchical storage. According to Patent Document 1, files that are frequently used are stored in the high-speed storage device such as a magnetic disk device, while files that are not frequently used are stored in the inexpensive, low-speed storage device such as a tape device. Which files should be placed, i.e., stored, in which storage device is determined by using a table that manages access frequency of each file.

[0003]

Disclosed in Patent Document 2 is a system, as typified by disk array subsystems, in which a plurality of logical storage devices having different processing speeds and storage capacities are configured within a storage device that is connected to a computer and used. The storage device discloses a technology to manage as statistical information the frequency of accesses from the computer to data stored in the storage device,





and, based on the statistical information, to transfer data with high access frequency to logical storage devices with higher performance.

[0004]

[Patent Document 1]

Patent Laid-Open Publication No. H9-297699 (Pg. 3 to 4, Fig. 1)

[Patent Document 2]

Patent Laid-Open Publication No. H09-274544 (Pg. 3, Figs. 6, 7)

[0005]

[Problem to be Solved by the Invention]

The first set of problems entailed in the prior art is that there is a high dependency on the computer connected to the storage device, that there is a limitation in the system configuration, and that it is difficult to simplify the system management.

[0006]

According to Patent Document 1, a hierarchical storage control is realized through software operating on the computer. The hierarchical storage control here refers to a data storage control for controlling a plurality of storage regions having different processing speeds and storage capacities such that the storage regions can be changed according to the frequency of data usage. In other words, the hierarchical storage control refers to controlling to select, according to the property of data such as frequency of data usage, an appropriate storage region from among a plurality of storage regions having different properties in terms of

The storage device according to claim 17, characterized in that, migrating the data of the file stored in said first storage region to said second storage region, said first interface control device controls to transmit to said second storage device through said third interface control device an access request containing an address of a storage region in said second storage region that stores the data of the file, and changes the relationship between identification information of the file and information indicating the storage region that stores the data of the file.

[Claim 19]

A storage device that is connected to a computer, the storage device comprising:

a first node that receives from the computer an access request having identification information of a file;

a second node that is connected to at least one first disk;

a third node that is connected to at least one second disk and a second storage device that is connected to the at least one second disk and has a file I/O interface control device that accepts an access request having identification information of a file;

a fourth node that is connected to at least one third disk and a third storage device that is connected to the at least one third disk and has a block I/O interface control device that accepts an access request having address information indicating a storage position of the data within the at least one third disk; and





a switch that mutually connects said first node, said second node, said third node and said fourth node,

and characterized in that a first storage region exists in said at least one first disk, a second storage region exists in said at least one second disk, and a third storage region exists in said at least one third disk, and

said first node controls to store the file in one of said first storage region, said second storage region and said third storage region according to property of the file specified by the identification information received from said computer.

[Claim 20]

The storage device according to claim 19, characterized in that,

when the data of the file is stored in said first storage region, said second node controls to store the data of the file in a storage region inside said at least one first disk, which is correlated to the identification information received from said computer,

when the data of the file is stored in said second storage region, said first node controls to transmit to said second storage device through said third node an access request containing identification information correlated to the file, and

when the data of the file is stored in said third storage region, said first node controls to transmit to said third storage device through said fourth node an access request containing address information indicating a storage position of the data of the file within said third storage region.

storage region.

[Claim 14]

A storage device that is connected to a computer, the storage device comprising:

a first interface control device that receives from said computer an access request having identification information for designating a file;

a second interface control device that is connected to said first interface control device;

a plurality of first disks that are connected to said second interface control device;

a third interface control device that is connected to a second storage device having a fourth interface control device that receives an access request containing address information indicating a storage position of data and having a plurality of second disks that are connected to the fourth interface control device;

a first storage region existing in said plurality of first disks; and

a second storage region existing in said plurality of second disks, and characterized in that

the first interface control device that received an access request from said computer decides as to which one of said first storage region and said second storage region to store data of the file according to property of the file indicated by identification information contained in the access request received,





when the data of said file is stored in said first storage region, said second interface control device stores the data of the file in one of said plurality of first disks, and

when the data of said file is stored in said second storage region, said first interface control device controls to transmit to said fourth interface control device through said third interface control device the access request containing address information within said second storage region where the data of the file is to be stored.

[Claim 15]

The storage device according to claim 14, characterized in that said third interface control device is an interface control device that corresponds to a block I/O interface.

[Claim 16]

The storage device according to claim 15, characterized in that said first interface control device sets up a file system in said second storage region.

[Claim 17]

The storage device according to claim 16, characterized in that said first interface control device controls to migrate data of a file from said first storage region to said second storage region through said third interface control device based on property of the file whose data is stored in said first storage region.

[Claim 18]

when the data of said file is stored in said second storage region, said first interface control device that received the access request from the computer controls such that the data of said file is transmitted to said second storage device through said first interface control device that is connected to said second computer.

[Claim 9]

The storage device according to claim 8, characterized in that

said second storage device includes a third interface control device that receives an access request having identification information of a file, and that accesses a storage region within said second storage region correlated to the identification information received to thereby access data of the file specified by the identification information,

said third interface control device sets a file system in said second storage region, and

the first interface control device that received an access request from said computer controls, when data of a file designated by identification information contained in the access request is stored in said second storage region, to transmit the access request having the identification information correlated to the file to said third interface control device through the first interface control device connected to said second storage device.

[Claim 10]

The storage device according to claim 9, characterized in that the first interface control device connected to said second storage device receives





an access request from said computer.

[Claim 11]

The storage device according to claim 9, characterized in that the first interface control device controls to migrate the data of the file from said first storage region to said second storage region through said third interface control device, based on property of files whose data is stored in said first storage region.

[Claim 12]

The storage device according to claim 11, characterized in that, when migrating data of a file stored in said first storage region to said second storage region, the first interface control device transmits to said third interface control device an access request having identification information correlated to the file, and stores the identification information of the file received from said computer correlated with the file system set in the second storage region.

[Claim 13]

The storage device according to claim 12, characterized in that

the first interface control device stores information concerning property of files and information concerning property of said first storage region and said second storage region, and decides, based on the information concerning property of files and the information concerning property of said first storage region and said second storage region, as to whether or not data of a file stored in said first storage region is to be migrated to said second

first storage region, and a second file system in said second storage region.

[Claim 5]

The storage device according to claim 4, characterized in that, according to static property and dynamic property of a file that is designated by the identification information received from the computer, static property being predetermined property and dynamic property being property that changes with passage of time since a point of time when the file is created, said first interface control device decides as to which one of said first storage region and said second storage region to store the data of the file indicated by said identification information.

[Claim 6]

The storage device according to claim 5, characterized in that said first interface control device controls to migrate data of a file stored in one of said first storage region and said second storage region to the other storage region according to a change in said dynamic property, and changes identification information that specifies the file and information that indicates correlation between the file and the storage position.

[Claim 7]

The storage device according to claim 6, characterized in that said static property includes information that specifies a type of the file, information that specifies a time when the file is created, or information that specifies a value of the file, and that said dynamic property includes information concerning an access property to the file, or information





concerning the time elapsed since the file is created.

[Claim 8]

A storage device that is connected to a computer, the storage device comprising:

at least one first interface control device that receives from said computer an access request containing identification information of a file;

at least one second interface control device connected to said at least one first interface control device; and

a plurality of first disks, each being connected to said at least one second interface control device,

and characterized in that one of said at least one first interface control device is connected to a second storage device having a plurality of second disks,

a first storage region is set in said plurality of first disks,

a second storage region is set in said plurality of second disks,

said first interface control device, upon receiving an access request from said computer, decides, according to property of a file designated by identification information contained in the access request received, as to which one of said first storage region and said second storage region to store data of the file,

when the data of said file is stored in said first storage region, said at least one second interface control device stores the data of the file in one of said plurality of first disks, and

[Document Name] Specification

[TITLE OF THE INVENTION]

STORAGE DEVICE

[Claims]

[Claim 1]

A storage device that is connected to at least one computer, the storage device comprising:

a first interface control device that receives from the computer an access request designating identification information of a file;

a second interface control device connected to said first interface control device; and

a plurality of disks connected to said second interface control device, and characterized in that said plurality of disks include at least one first disk, and at least one second disk, the first disk and the second disk being different kinds,

said first interface control device decides, based on identification information received from the computer, a storage position of data of the file designated by the identification information within said plurality of disks, and that

said second interface control device controls to store the data of the file designated by said identification information at the storage position decided by said first interface control device.

[Claim 2]





The storage device according to claim 1, characterized in that said first disk is a Fiber Channel disk equipped with a Fiber Channel type interface, and said second disk is a serial ATA disk equipped with a serial ATA type interface.

[Claim 3]

The storage device according to claim 1, further comprising a memory, a memory controller for controlling said memory, a plurality of first interface control devices, each being connected to the memory controller, and a plurality of second interface control devices, each being connected to said memory controller,

characterized in that the first interface control devices that received identification information of a file and data of the file from the computer stores the data of the file in the memory, and that

one of said second interface control devices that is connected to one of the disks in which the data of said file is stored controls to store the data of said file stored in said memory in the one of the disks according to the storage position of the data of said file decided by said first interface control device.

[Claim 4]

The storage device according to claim 1, characterized in that

a first storage region exists in the at least one first disk,

a second storage region exists in the at least one second disk, and

said first interface control device sets up a first file system in said

[Document Name] Patent Application

[Reference Number] K03001941A

[Addressee] Director-General of the Patent Office

[International Classification] G06F 12/00

[Inventor]

[Address] c/o System Development Laboratory, Hitachi Ltd.
1099 Ozenji, Aso-ku, Kawasaki-shi, Kanagawa-ken

[Name] Naoto Matsunami

[Inventor]

[Address] c/o System Development Laboratory, Hitachi Ltd.
1099 Ozenji, Aso-ku, Kawasaki-shi, Kanagawa-ken

[Name] Koji Sonoda

[Inventor]

[Address] c/o System Development Laboratory, Hitachi Ltd.
1099 Ozenji, Aso-ku, Kawasaki-shi, Kanagawa-ken

[Name] Akira Yamamoto

[Inventor]

[Address] c/o Storage Division, Hitachi Ltd.
2880 Kozu, Odawara-shi, Kanagawa-ken

[Name] Masafumi Nozawa

[Inventor]

[Address] c/o System Development Laboratory, Hitachi Ltd.
1099 Ozenji, Aso-ku, Kawasaki-shi, Kanagawa-ken

[Name] Masaaki Iwasaki

[Applicant]

[ID Number] 000005108

[Name] Hitachi Ltd.

[Attorney]

[ID Number] 100075096

[Patent Attorney]





[Name] Yasuo Sakuta

[Fee]

[Deposit Account Number] 013088

[Amount of Deposit] 21,000 yen

[List of Documents Filed]

[Name of Document] Specification 1

[Name of Document] Drawing 1

[Name of Document] Abstract 1

[Requirement of Proof] Required